Wind-Driven Snow Buildup Using a Level Set Approach

Tommy Hinks[1†] and Ken Museth[2‡]

[1]University College Dublin, Ireland
[2]DreamWorks Animation

Abstract

We present a physically-based snow modeling approach that handles geometrically complex scenes and arbitrary amounts of accumulated snow. Scene objects are represented with a novel dual level set structure. This implicit surface representation produces smooth snow surfaces that adhere to granular stability constraints at every time-step. Realistic accumulation patterns are achieved by tracing snow-carrying particles in a dynamic wind-field and on the surfaces of objects. Local level set operations are used to deposit snow at surface locations for which accumulation is physically plausible. The effectiveness of our method is demonstrated by applying our method to a number of challenging scenes.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.6]: Simulation and modeling—Computer Graphics [I.3.5]: Physically based modeling—

1. Introduction

Snow is common during the winter season in many parts of the world. Heavy snowfall upon a scene will dramatically change its appearance in terms of shape as well as illumination. The granular nature of snow allows it to completely cover small objects and accumulate on sharp features, smoothing them. Snow is arguably one of nature's most complex and fascinating substances, and finding accurate models for this phenomenon has proven to be a great challenge to the scientific community for many years [Nak54].

We strongly believe that the computer graphics industry would benefit from a robust method for snow distribution that is independent of the scene complexity. Such techniques could be used to generate snow-covered scenes for movies as well as interactive applications.

The two major issues to resolve in order to generate realistic snow scenes are: (1) Assuring that snow accumulates at correct locations, taking into account external factors such as wind; (2) Guaranteeing that snow accumulates in a way that is physically plausible with respect to granular stability.

† tommy.hinks@ucd.ie
‡ ken.museth@dreamworks.com

Our general approach is based on the concept of *snow packages*. Snow packages represent discrete volumes of snow and are traced in a wind-field produced from fluid solver and on the surfaces of objects. We present an implicit surfacing approach for modeling of progressive snow accumulation on static surfaces. Our contributions can be summarized as follows: (1) *Dual level set* structures are used to represent the surfaces of the dynamic snow and the static boundaries of a scene, allowing for topologically complex modeling of snow. Additionally, the signed distance field representations of the level sets accelerate fluid voxel classification, as well as providing cheap closest-point and normal computations; (2) Particle-level set interaction through local level set operations are used to deposit stable amounts of snow on surfaces, according to a physically-based stability criterion; (3) Our snow model is based on a steady-state description where each frame represents a snapshot of the physically-based buildup. This lets us interactively monitor the buildup process and interrupt it at any moment.

The structure of this paper is as follows: In Section 2 we discuss previous techniques for modeling snow; Section 3 gives an overview of level set methods; Section 4 explains the details of our approach; Section 5 demonstrates the effectiveness of our method and compares our results with previous techniques; Section 6 summarizes our results and gives some possible directions for future work.

2. Related Work

Earlier work in snow modeling can be divided into physically and non-physically based approaches. Our method belongs to the former category. The latter category generates snow-covered scenes using occlusion mapping techniques [PTS99, Dud05, FB07]. Accumulated snow is represented through texture-mapping or simple displacement techniques. While such methods produce reasonable results for large terrain scenes, they are not convincing for detailed models or close-up views. Also, these methods are typically not capable of incorporating wind effects.

Physically based approaches differ from each other in two major aspects: (1) Geometric representation of objects in the scene; (2) Snow transportation mechanisms. Height fields were used in [SOH99, FO02], and while it is intuitive to accumulate snow on such representations, it imposes severe limitations on the types of scenes that can be modeled. Height span maps were used by [ON05, vFG09] in an attempt to overcome these limitations. However, the snow distribution method of the former is aimed at real-time applications and lacks fine detail, while the latter relies purely on statistical models and is not capable of simulating wind-driven accumulation patterns.

Polygon meshes were first used by [Fea00a], and later on by [MMAL05]. Intricate subdivision schemes are required in order to increase the level-of-detail in areas with complex occlusion patterns. Also, their methods require a global refinement step to handle unstable accumulation configurations, involving costly height sorting of polygons where snow must be transported to lower areas. A direct consequence of the necessity for a final step is that snow buildup cannot be animated over time, requiring users to run the simulation to the very end before being able to evaluate the results. Further, explicit geometric representations tend to produce sharp edges for accumulated snow, whereas in reality granular buildup tends to be very smooth. An implicit surface approach using metaballs was proposed in [NIDN97] and produces smooth surfaces. However, the positioning of the metaballs was done manually and the accumulation patterns are not convincing.

Fearing [Fea00a] used an importance sampling scheme for particle tracing through a static wind-field with a globally set wind-direction. Particles are emitted from upward-facing surfaces towards the sky to determine exposure to falling snow, where the amount of exposure determines the amount of snow accumulating on the surface. Using a global wind-direction has the advantage of not requiring any fluid computations, but does not produce fully convincing accumulation patterns. Snow transportation methods using dynamic wind-fields, computed using the objects in the scene as internal boundaries, have been proposed. In [FO02] snow is transported in a dynamic wind-field using a density convection approach [FSJ01]. Notably, this is the only method that allows accumulated snow to be redistributed in the scene. A model for snowflake motion was introduced by [MMAL05] and produces realistic looking animations of snowfall. This model was used to trace particles from the sky downwards towards the scene, thereby providing exposure information similar to that in [Fea00a].

In contrast to previous work our approach uses a level set surface representation that is inherently smooth, and allows arbitrary amounts of snow to be accumulated. Our snow transportation method is inspired by that of [MMAL05], with the major difference that we trace quantized volumes of snow as if they were single snowflakes. The volumes are deposited on surfaces through local operations. We enforce that the accumulated snow is always be in a stable configuration, thereby avoiding the need for a global refinement step at the end. Previous work in volumetric texturing using level sets includes [BMPB08], and focuses on the blending of general shapes rather than systematic volumetric buildup through a large number of interactions. Before we present the details of our approach we review the level set methods used next.

3. Level Set Methods

Level set methods have been successfully applied to interface (i.e. surface) tracking problems in computer graphics, computer vision and computational physics [OS88]. Level sets are implicit surface representations, making them well-suited for problems where the surface topology changes arbitrarily over time. An static implicit surface can be defined as

$$S \equiv \phi^{-1}(0) \equiv \{\vec{x} \,|\, \phi(\vec{x}) = 0\} \tag{1}$$

where $\vec{x} \in \Re^3$ is a point in space. Thus, the iso-surface is implicitly defined as the set of points that solve the equation $\phi(\vec{x}) = 0$. Further, let $\phi(\vec{x})$ be a signed Euclidean distance function, such that the value of $\phi(\vec{x})$ is always the closest distance to the interface, satisfying the following two conditions

$$\begin{aligned} \phi(\vec{x}) &= \min(\|\vec{x} - \vec{x}_s\|) \; \forall \, \vec{x}_s \,|\, \phi(\vec{x}_s) = 0 \\ \|\nabla\phi\| &= 1 \end{aligned}$$

where the gradient is defined as

$$\nabla\phi \equiv \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\right)$$

For the interior region Ω bound by S the following rules are used

$$\phi(\vec{x}) < 0 \rightarrow \vec{x} \in \Omega \tag{2}$$

$$\phi(\vec{x}) > 0 \rightarrow \vec{x} \notin \Omega \tag{3}$$

$$\phi(\vec{x}) = 0 \rightarrow \vec{x} \in \partial\Omega \equiv S \tag{4}$$

Points inside Ω have negative distances to the surface, while points outside Ω have positive distances. Thus, a point $\vec{x}$ can be tested for membership in Ω simply by checking the sign of $\phi(\vec{x})$. As for all iso-surfaces, the gradient direction for a

point on the surface is perpendicular to the tangent plane at that point. Thus, the unit normal for a point on the surface is defined as

$$\hat{n} \equiv \frac{\nabla\phi(\vec{x}_s)}{\|\nabla\phi(\vec{x}_s)\|} \tag{5}$$

Further, the closest point $\vec{x}_s$ on the interface for any point $\vec{x}$ in the domain of ϕ can be found using the *closest point transform* (CPT)

$$\vec{x}_s = \vec{x} - \phi(\vec{x})\nabla\phi(\vec{x}) \tag{6}$$

In order to allow the surface to change shape over time, the static interface definition (Equation 1) is made time-dependent

$$S(t) \equiv \phi^{-1}(0,t) \equiv \{\vec{x}(t) \,|\, \phi(\vec{x}(t),t) = 0\} \tag{7}$$

A *speed function*, $F_v(\vec{x},\hat{n},...)$, is used to *propagate* the interface in the normal direction over time by modifying the distance field [OF02]. The speed function can depend on any number of arguments, but must always return a scalar value. In general, propagation invalidates the Euclidean distance field, which then needs to be *renormalized* by ensuring that the Eikonal equation $\|\nabla\phi\| = 1$ is satisfied [Set99]. It is desirable that the surface remains in place during renormalization. Therefore, distance information is assumed to be correct at the surface and propagated outwards using fast marching methods [Zha04].

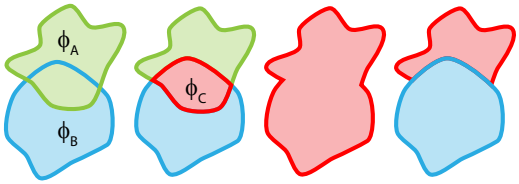

Figure 1: *CSG operators, from left: Two distance fields, intersection, union, difference ($\phi_A - \phi_B$).*

Constructive Solid Geometry (CSG) is a technique for combining shapes using boolean operations. CSG operators can be elegantly implemented using level sets [MBWB02]. Three basic boolean operations are shown in Figure 1 and the basic formulations are given in Table 1.

Table 1: *CSG Boolean Operations*

Operator	Notation	Implementation
Intersection	$\phi_C = \phi_A \bigcap \phi_B$	$\phi_C = \max(\phi_A, \phi_B)$
Union	$\phi_C = \phi_A \bigcup \phi_B$	$\phi_C = \min(\phi_A, \phi_B)$
Difference	$\phi_C = \phi_A - \phi_B$	$\phi_C = \max(\phi_A, -\phi_B)$

For computational purposes the distance field ϕ is discretized using some type of grid structure, where each voxel stores the signed Euclidean distance to the surface at its center. Distances at non-integer coordinates are obtained through trilinear interpolation. Since distances close to the interface are of most interest, maintaining a grid for the entire volume spanned by the interface is highly inefficient, both in terms of memory and computational cost. Instead, we use the compact *dynamic tubular grid* (DT-Grid) structure [NM06], which stores voxels in a narrow band around the interface only. Also, the narrow band is rebuilt dynamically, allowing the interface to propagate outside the original bounding volume, which in our case means that an arbitrary amount of snow can be accumulated in a scene without further complications.

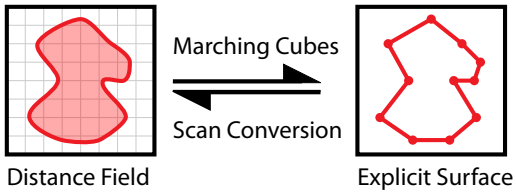

Figure 2: *Conversion between distance fields and explicit surfaces.*

Acquiring level set representations from explicit surface representations (e.g. triangle meshes) is done through a procedure known as *scan conversion* [Mau03]. Currently, most scan conversion algorithms require water-tight meshes to produce reliable results. However, water-tight models are typically desirable for most purposes and impose no serious limitation on our method. For rendering purposes it may be practical to convert a level set into a triangle mesh. This is done using the *marching cubes* algorithm [LC87]. The relationship between these two surface representations is shown in Figure 2.

4. Snow Modeling

Our scenes consist of one or more solid objects represented using a novel *dual level set* structure. Each object stores two level sets, a static level set (ϕ_S) for the initial solid surface, and a dynamic snow level set (ϕ_D) used to track built up snow. At the end of the simulation the built up snow surface corresponds to the boolean difference between the dynamic and static level sets: $\phi_{snow} = \phi_D - \phi_S$. Initially, the two level sets are identical (Figure 3), corresponding to a complete absence of snow on the object. For the duration of the simulation ϕ_D represents both the solid object and built up snow. The distance fields are stored on DT-Grids [NM06], which allows us to use detailed models and significantly reduces computation times.

A fluid domain is defined such that it fully contains the scene. Cells on the boundary of the fluid domain are initialized using a global wind velocity. Velocities within the domain are solved for using a standard Navier-Stokes solver [Sta99]. For locations outside the fluid domain the global wind velocity is used. Internal fluid cells are classified as solid if the cell center is inside any object, otherwise

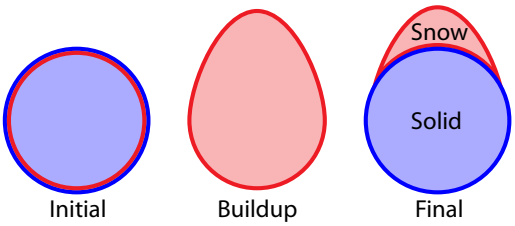

Figure 3: *Initially, the dynamic and static surfaces are identical. During snow buildup the dynamic interface is propagated to represent deposited snow. In a final step, built up snow is extracted using the CSG difference operator.*

as empty. Note that the inside-test simply requires transforming the cell center to object coordinates and checking for negative distances in ϕ_D (Equation 2). It is not necessary to check ϕ_S, since during the simulation $\phi_D \leq \phi_S$ for all points in space. In contrast, such testing using polygon meshes is non-trivial and inevitably more time-consuming. During the simulation, the wind field is recomputed whenever a change in internal fluid cell classification occurs.

Our snow transportation model is based on the concept of *snow packages*, which we define as particles transporting discrete volumes of snow. Particles are advected in the velocity field generated by the fluid solver and on surfaces in the scene. Collisions between particles and dynamic level sets cause the transported volumes of snow to be deposited in the scene by localized propagations. Next, we present the details of our stability criterion, which allows us to accumulate snow in realistic configurations. Following that, we present our model for snow transportation and how snow is deposited on surfaces.

4.1. Snow Stability

Intuitively, granular buildup does not have sharp edges. Friction between particles results in attractive forces, allowing particles to be stacked and form a volume. An external force, such as gravity, may cause particles at the volume boundary to slide off the volume. The sliding (e.g. avalanches) continues until a stable, and smooth, configuration is reached, where frictional and external forces are balanced. However, modeling the frictional forces inside snow volumes is non-trivial. Complex geometric interactions between fractal-like snowflakes, as well as internal pressure variations and melting, make snow an extremely challenging substance to simulate accurately [GM81, MC00].

However, for visualization purposes, we can simulate snow stability without explicitly modeling frictional forces. We model stability in terms of surface orientation and our stability criterion depends only on a global temperature value. For any point $\vec{x}_s$ on the snow surface we can compute an angle θ between the surface normal $\hat{n}$ (Equation 5) and a (horizontal) plane π_H. The plane π_H is perpendicular

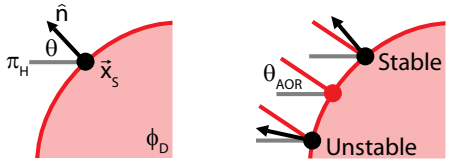

Figure 4: *Left: Snow stability is measured using the angle θ between the surface normal $\hat{n}$ and a horizontal plane π_H. Right: A point on the surface is stable if $\theta \geq \theta_{AOR}$, otherwise unstable.*

to a (vertical) gravitational direction $\vec{g}$ and passes through $\vec{x}_s$ (Figure 4). Further, we say that a point $\vec{x}_s$ is *stable* if θ is larger than some *angle of repose*, θ_{AOR}. Stable points on dynamic level set surfaces are candidate locations for snow buildup, as will be further explained in Section 4.4.

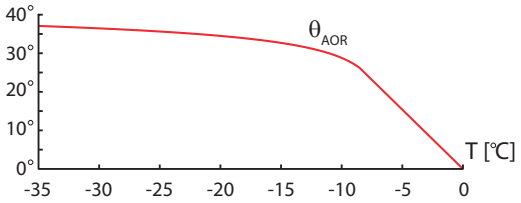

Figure 5: *Angle of repose as a function of temperature.*

As an adaptation of the experimental work by Fearing [Fea00b], we compute a temperature-dependent θ_{AOR} as

$$\theta_{AOR} = \begin{cases} 50 - 30(|T+6|^{-0.25}) & T \in [-35, -8.5) \\ -\frac{26.1418}{8.5} T & T \in [-8.5, 0] \end{cases}$$

where T is the temperature in $^{\circ}C$ and θ_{AOR} is the angle of repose in degrees. This model is valid for $T \in [-35, 0]$ and the curve is shown in Figure 5. The physical interpretation of θ_{AOR} is that colder snow is more "powdery", with less frictional interaction between particles. A manifestation of small θ_{AOR} is that snow accumulates on steeper surfaces. Before we explain how dynamic level sets are propagated with adherence to our stability criterion, we present our snow transportation model next.

4.2. Snow Packages

Two types of spherical particles, or *snow packages*, are used in our snow transportation model: (1) *wind packages* – that are advected in the wind-field using a simplified version of the snowflake motion model from [MMAL05], ignoring rotation; (2) *slide packages* – that are traced on surfaces, depositing snow at stable locations. Simple first-order Euler time-integration has been found to be sufficiently accurate for snow package motion.

Wind packages with a user-defined radius are initialized at random locations on a parameterized patch located above the

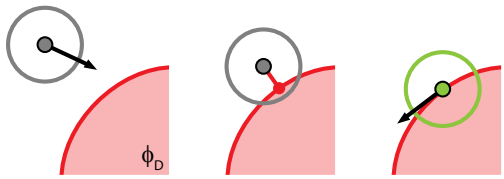

Figure 6: *Left: Wind packages are traced in the wind-field. Middle: When a wind package collides with a snow surface it is centered on the surface using the CPT. Right: The collided wind package is converted into a slide package.*

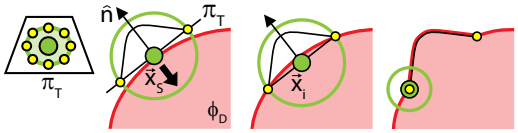

Figure 7: *Left: A slide package at a stable surface point $\vec{x}_s$ is moved inwards. Middle: The updated position $\vec{x}_i$ is found by examining ϕ_D at boundary points (yellow). Right: The surface is propagated and a smaller slide package is created from any significant remaining package volume.*

scene. Snow distribution in the scene is determined by collisions between wind packages and level set surfaces. A collision between a wind package and a surface causes the wind package to be converted into a slide package, positioned on the surface using the CPT (Equation 6), as illustrated in Figure 6. Slide packages are traced separately along the projection of the gravity vector onto the surface tangent plane. Before and after every time-step we guarantee that slide packages are on the surface using the CPT. Should the closest point on the surface be downward-facing, the slide package is nudged away from the surface and converted back into a wind package. In the case where the closest point is a stable location on the surface, the package volume is added to built up snow through a level set propagation, explained in more detail in Section 4.4. Finally, a snow package is removed if it can no longer collide with surfaces in the scene. Next, we present further details on collision detection, before introducing methods for transferring slide package volumes into built up snow.

4.3. Collision Detection

Since our surfaces are represented by distance fields it becomes trivial to test for collisions with wind packages. We simply interpolate a value for the distance field at the wind package position ($\vec{x}_{wp}$) and if this value is smaller than the wind package radius (r_{wp}) a collision has occurred. Thus, if $|\phi_D(\vec{x}_{wp})| \leq r_{wp}$ the wind package has collided with a snow surface. Recall that DT-Grids store distances only inside a narrow band around the surface. This implies that the width γ of the narrow band should be set so that $\gamma \geq \max(r_{wp})$. Next, we introduce methods for converting slide package volumes into built up snow.

4.4. Snow Buildup

As a slide package moves on a level set surface, the transported snow is deposited by propagating the surface outwards. The spherical domain of the level set propagation is bound by the slide package radius. An important postcondition for the propagation is that all surface points within the domain are stable.

At each time-step, a slide package's position $\vec{x}_s$ on the surface is checked for stability (Section 4.1). For an unstable position no propagation is performed and the package simply moves on. However, for a stable position the package is moved inwards along the negative normal direction to a new position $\vec{x}_i$ (Figure 7). The updated position is found by examining distances in ϕ_D at boundary points along the circle of intersection between the tangent plane π_T and the spherical package domain. The package is iteratively moved inwards by the largest positive distance d_{max} until $d_{max} \leq 0$. It is necessary to check that $\phi_D(\vec{x}_i) \leq 0$, to verify that the slide package is still on the inside of ϕ_D. Similarly, if $|\phi_D(\vec{x}_i)| \geq \|\vec{x}_s - \vec{x}_i\|$ the slide package domain does not contain the original surface point $\vec{x}_s$. In both cases the slide package is moved on without triggering a propagation.

Next, the level set surface within the domain is propagated to match a *shape function*, defined such that the surface within the domain is unconditionally stable. The shape function is discussed in more detail in Section 4.5. Any significant slide package volume not represented by the propagation is converted into a new slide package initialized at the lowest boundary point.

Propagation is done by iteratively moving the surface within the domain outwards in small increments until points on the level set surface lie on the curve specified by the shape function. While this preserves the distance field inside the domain, it becomes invalid for points outside, making it necessary to perform a computationally costly reinitialization of ϕ_D. Therefore, instead of triggering each propagation immediately, followed by a reinitialization, propagations are stored in a *propagation buffer* with a certain size (we use 128). When the buffer becomes full, the propagations are carried out in sequence, followed by a single reinitialization, thereby saving computational time. Before presenting our results, we provide the details of our shape function in the following section.

4.5. Shape Function

Our shape function is defined to guarantee unconditional surface stability within any propagation domain. Below, we present the details of our shape function.

Our shape function is of the form $F_s(d)$, where d is the distance to $\vec{x}_i$ in the tangent plane π_T (Figure 8). As such,

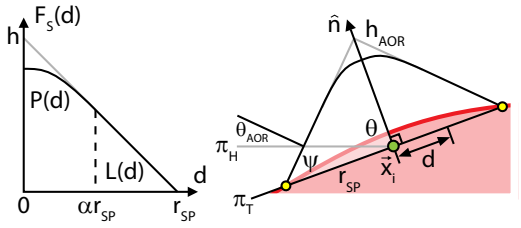

Figure 8: *Left: Basic form of our shape function. Right: The shape function is radially symmetric around the normal $\hat{n}$. We compute a stable height h_{AOR} from the trigonometric relationships shown.*

$F_s(d)$ is radially symmetric around $\hat{n}$. More precisely, $F_s(d)$ is defined as

$$
\begin{aligned}
F_s(d) &= \begin{cases} P(d) & d \in [0, \alpha r_{sp}) \\ L(d) & d \in [\alpha r_{sp}, r_{sp}] \\ 0 & d \in (r_{sp}, \infty) \end{cases} \\
P(d) &= -(\frac{h}{2\alpha r_{sp}^2})d^2 + h(1 - \frac{\alpha}{2}) \\
L(d) &= -\frac{h}{r_{sp}}d + h
\end{aligned}
$$

where r_{sp} is the slide package radius and $\alpha \in (0,1]$ is a smoothing parameter, governing not only the transition between $P(d)$ and $L(d)$, but also the shape of $P(d)$, see Figure 8. A small α gives a sharp linear shape, while a larger α gives a smoother parabolic shape. The parameter h determines the height of the curve, and it is shown below how this parameter is used to enforce the stability criterion.

Since updating the level set is a computationally expensive operation, it is desirable to add as much snow volume as possible per propagation. Thus, we wish to find the maximum stable height h_{AOR} for which every point on the curve is stable. Starting from the inner angle ψ (Figure 8), we can derive a height for which the steepest part of the curve, i.e. $L(d)$, is stable. We have that $h_{AOR} = r_{sp}\tan\psi$. Using the relation $\psi = \theta - \theta_{AOR}$, we find that $h_{AOR} = r_{sp}\tan(\theta - \theta_{AOR})$. Note that h_{AOR} may need to be clamped to r_{sp}, so that the shape fits in the domain.

Because $F_s(d)$ is radially symmetric around $\hat{n}$, the volume V_s enclosed between $F_s(d)$ and π_T is given by the solid of revolution for $F_s(d)$

$$
\begin{aligned}
V_s(h_{AOR}, r_{sp}, \alpha) &= 2\pi \int_0^{\alpha r_{sp}} P(x)x\,dx + 2\pi \int_{\alpha r_{sp}}^{r_{sp}} L(x)x\,dx \\
&= \pi h_{AOR} r_{sp}^2 C_\alpha \\
C_\alpha &= [\alpha^2(1 - \frac{3\alpha}{4}) + \frac{1}{3}(1 + \alpha^2(2\alpha - 3))]
\end{aligned}
$$

Thus, V_s approximates the volume added through propagation, ignoring here that parts of the tangent plane may already be inside the surface. In most cases $\vec{x}_i$ is close to the surface and for small domains the initial surface tends to be relatively flat. We note that, as expected, for $\alpha = 0$ the above expression simplifies into the well-known volume formula for a cone. Finally, we compute the remaining volume ΔV by subtracting V_s from the spherical package volume:

$$\Delta V = \frac{4\pi r_{sp}^3}{3} - V_s(h_{AOR}, r_{sp}, \alpha)$$

The following section present results using our method and a comparison with two related techniques.

5. Results

This section demonstrates the effectiveness of our method by applying it to four scenes. Results were generated using an unoptimized C++ implementation of our method running on a 2.0 GHz single-core machine. Scenes were rendered with a commercial ray-tracing package and a simple Lambertian shader was used for the snow surfaces. All scenes were represented as a single level set and $\alpha = 0.8$ was used throughout. Our first three tests (Figures 9–11) focus on snow buildup and do not use a wind-field. The fourth and final test scene (Figure 12) uses a wind-field and results are compared with two existing techniques.

Table 2: *Scene Settings Used in Tests*

	Res.	r_{wp}	Temp.	Time
Figure 9	128^3	$0.05m$	$-\{2,8\}^\circ C$	~$1.5h$
Figure 10	128^3	$0.10m$	$-5^\circ C$	~$3h$
Figure 11	256^3	$0.02m$	$-5^\circ C$	~$4h$
Figure 12	128^3	$0.10m$	$-3^\circ C$	~$4h$

Table 2 shows the settings used in our tests. The first column shows the effective level set resolution used, the second is the radius of emitted wind packages. Temperature is shown in the third column and the last column gives the time spent on simulation.

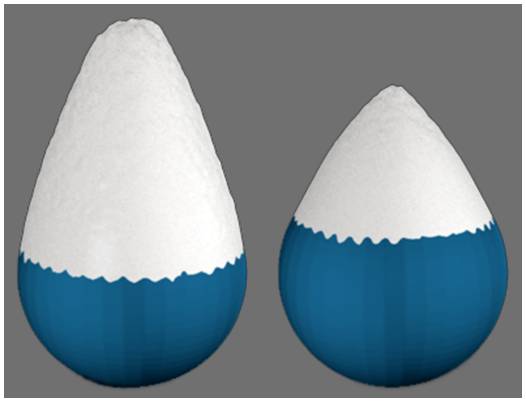

Figure 9: *Snow buildup on spheres using different temperatures. Left: $-2^\circ C$. Right: $-8^\circ C$.*

Figure 9 shows snow buildup on two spheres with unit radius using different temperatures. The figure illustrates well

the effect of θ_{AOR}, where a smaller θ_{AOR} (left sphere) allows snow to accumulate in a steeper configuration. Notice how snow surfaces have propagated beyond the initial bounding volume of each sphere.

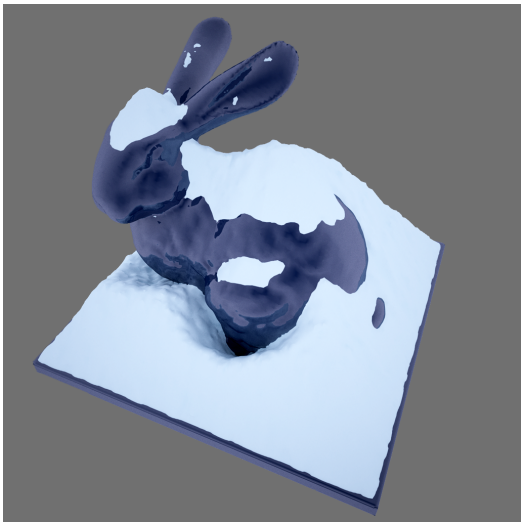

Figure 10: *Large amount of snow accumulated on the Stanford bunny. (Model courtesy of the Stanford 3D Scanning Repository)*

Figure 10 shows a large amount of snow accumulated on a scene with a Stanford Bunny resting on a plane. The scene was scaled such that the bounding volume was one meter. Our implicit surfaces automatically adapt to the topological changes where built up snow on the plane joins that on the back of the bunny.

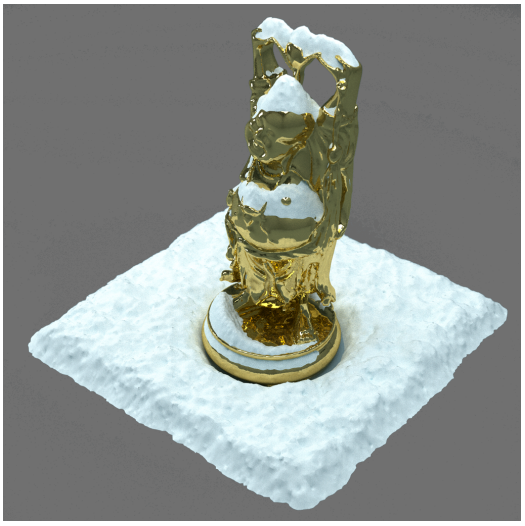

Figure 11: *Snow buildup on detailed, geometrically complex model. (Model courtesy of the Stanford 3D Scanning Repository)*

The detailed model shown in Figure 11 is challenging for existing methods due to the large amount of geometric overlap in the gravitational direction. The statue is $0.2m$ high. As shown, our method naturally extends to arbitrarily complex models.

A simple scene, consisting of three box-like buildings on a plane, first used in [FO02], was used to compare our method with previous wind-driven snow buildup techniques (Figure 12). Our version of this scene is 10 by 10 meters in the horizontal. Fluid solver resolution was 64^3 and wind speed at the boundaries of the fluid domain were $1m/s$, flowing left-to-right in the image. Our scene was generated in 4 hours, compared to 3 hours reported in [FO02] (no information available for [MMAL05]). Hardware details were omitted from [FO02], but the setup we used is similar in raw performance. Although our method is slower at present, it has the distinct advantage of not being limited to height fields. Snow accumulation patterns are similar in the three scenes and variations are, to some extent, caused by slight differences in setup and scene representations. However, results are still visually comparable, and our method generates plausible accumulation patterns, as well as extending naturally to more complex scenes.

6. Conclusions and Future Work

We have presented a snow buildup method suitable for geometrically complex scenes. Surfaces are represented using level sets, efficiently stored on compact DT-Grids. A particle approach is used to transport snow volumes, allowing us to simulate wind-driven snow buildup. Localized level set operations are used to deposit stable snow volumes at physically plausible locations.

A missing feature in our method is the redistribution of built up snow. For this to be added, a model for removing snow such that remaining volumes are stable is required. Possibly, a snow transportation model using density fields is suitable for this purpose. However, it would require new methods for interaction between level sets and density fields. We also predict that significant optimizations, with regards to level set operations, can be made to reduce simulation times. In particular, a significant part of reinitialization computations are spent on unchanged, or conservatively propagated voxels. Finally, we would consider snow buildup based solely on stable surface points, ignoring any transportation aspects. This would drastically increase performance and would be suitable for scenes with simple occlusion conditions, where large volumes of deposited snow are required.

7. Acknowledgments

Special thanks to Michael Bang Nielsen for providing an implementation of the DT-Grid, Andreas Söderström for helping with the fluid solver, and Ola Nilsson for many insightful comments.

Figure 12: *Left: Original scene from [FO02], results are smooth but the method is limited to height fields. Middle: [MMAL05] use polygon meshes and the accumulated snow has sharp, uncharacteristic edges. Right: Our method produces smooth buildup and extends to complex geometry.*

References

[BMPB08] BRODERSEN A., MUSETH K., PORUMBESCU S., BUDGE B.: Geometric texturing using level sets. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 277–288.

[Dud05] DUDASH B.: *Snow Accumulation*. Tech. rep., NVIDIA, 2005.

[FB07] FOLDES D., BENES B.: Occlusion-based snow accumulation simulation. In *Vriphys '07* (2007), pp. 35–41.

[Fea00a] FEARING P.: Computer modelling of fallen snow. In *SIGGRAPH '00* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 37–46.

[Fea00b] FEARING P.: *The Computer Modelling of Fallen Snow*. PhD thesis, UBC, 2000.

[FO02] FELDMAN B. E., O'BRIEN J. F.: Modeling the accumulation of wind-driven snow. In *SIGGRAPH '02: conference abstracts and applications* (New York, NY, USA, 2002), ACM, pp. 218–218.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH '01* (New York, NY, USA, 2001), ACM, pp. 15–22.

[GM81] GRAY D. M., MALE D. H.: *Handbook of snow: principles, processes, management and use*. Pergamon Press, 1981.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169.

[Mau03] MAUCH S.: *Efficient algorithms for solving static hamilton-jacobi equations*. PhD thesis, California Institute of Technology, 2003.

[MBWB02] MUSETH K., BREEN D. E., WHITAKER R. T., BARR A. H.: Level set surface editing operators. *ACM Trans. Graph. 21*, 3 (2002), 330–338.

[MC00] MURAOKA K., CHIBA N.: Visual simulation of snowfall, snow cover and snowmelt. In *ICPADS '00* (Washington, DC, USA, 2000), IEEE Computer Society, pp. 187–194.

[MMAL05] MOESLUND T. B., MADSEN C. B., AAGAARD M., LERCHE D.: Modeling falling and accumulating snow. In *Vision, Video and Graphics* (July 2005), pp. 61–68.

[Nak54] NAKAYA U.: *Snow Crystals: Natural and Artificial*. Harvard University Press, 1954.

[NIDN97] NISHITA T., IWASAKI H., DOBASHI H., NAKAMEI E.: A modeling and rendering method for snow by using metaballs. *Computer Graphics Forum 16*, 3 (1997), 357–364.

[NM06] NIELSEN M. B., MUSETH K.: Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *J. Sci. Comput. 26*, 3 (2006), 261–299.

[OF02] OSHER S., FEDKIW R.: *Level Set Methods and Dynamic Implicit Surfaces*. Springer New York, 2002.

[ON05] ONOUE K., NISHITA T.: An interactive deformation system for granular materials. *Computer Graphics Forum 24*, 1 (2005), 51–60.

[OS88] OSHER S., SETHIAN J. A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *J. Comput. Phys. 79*, 1 (1988), 12–49.

[PTS99] PREMOZE S., THOMPSON W. B., SHIRLEY P.: Geospecific rendering of alpine terrain. In *Eurographics Workshop on Rendering* (1999), Springer-Verlag, pp. 107–118.

[Set99] SETHIAN J. A.: *Level Set Methods and Fast Marching Methods*, 2 ed. Cambridge University Press, 1999.

[SOH99] SUMNER R., O'BRIEN J., HODGINS J.: Animating sand, mud and snow. *Computer Graphics Forum 18*, 1 (1999), 17–26.

[Sta99] STAM J.: Stable fluids. In *SIGGRAPH '99* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

[vFG09] V. FESTENBERG N., GUMHOLD S.: A geometric algorithm for snow distribution of snow in virtual scenes. *Eurographics Workshop on Natural Phenomena* (2009), 9.

[Zha04] ZHAO H.: Fast sweeping method for eikonal equations. *Mathematics of Computation 74* (2004), 603–627.